CRAB: Introduction

CRAB tutorial
20 June 2008

Eric Vaandering

CMS/Fermilab

CMS and the GRID

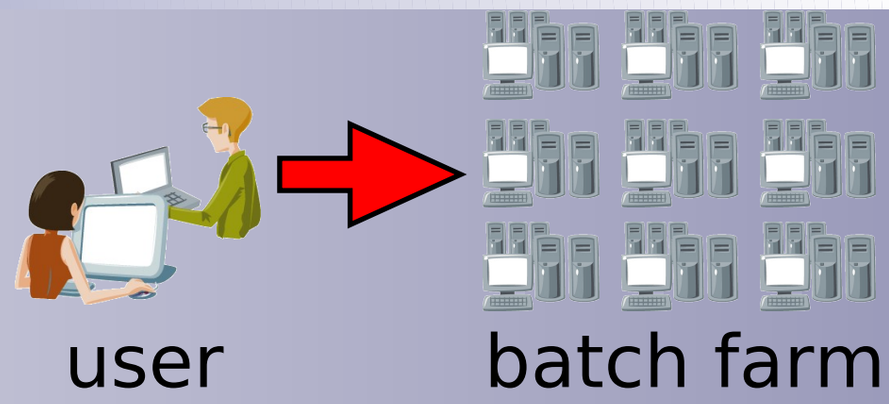
- The GRID
  - Interconnects computing resources (batch farms, storage systems, ...) in a transparent way
  - Provides every user with access to these resources
    - Authorization, job submission, ...
- CMS computing model distributes data world-wide
  - CMS will use the GRID to provide data access for all collaborators. User analysis at Tier2.
- CRAB (CMS Remote Analysis Builder) is the CMS user front end to the GRID

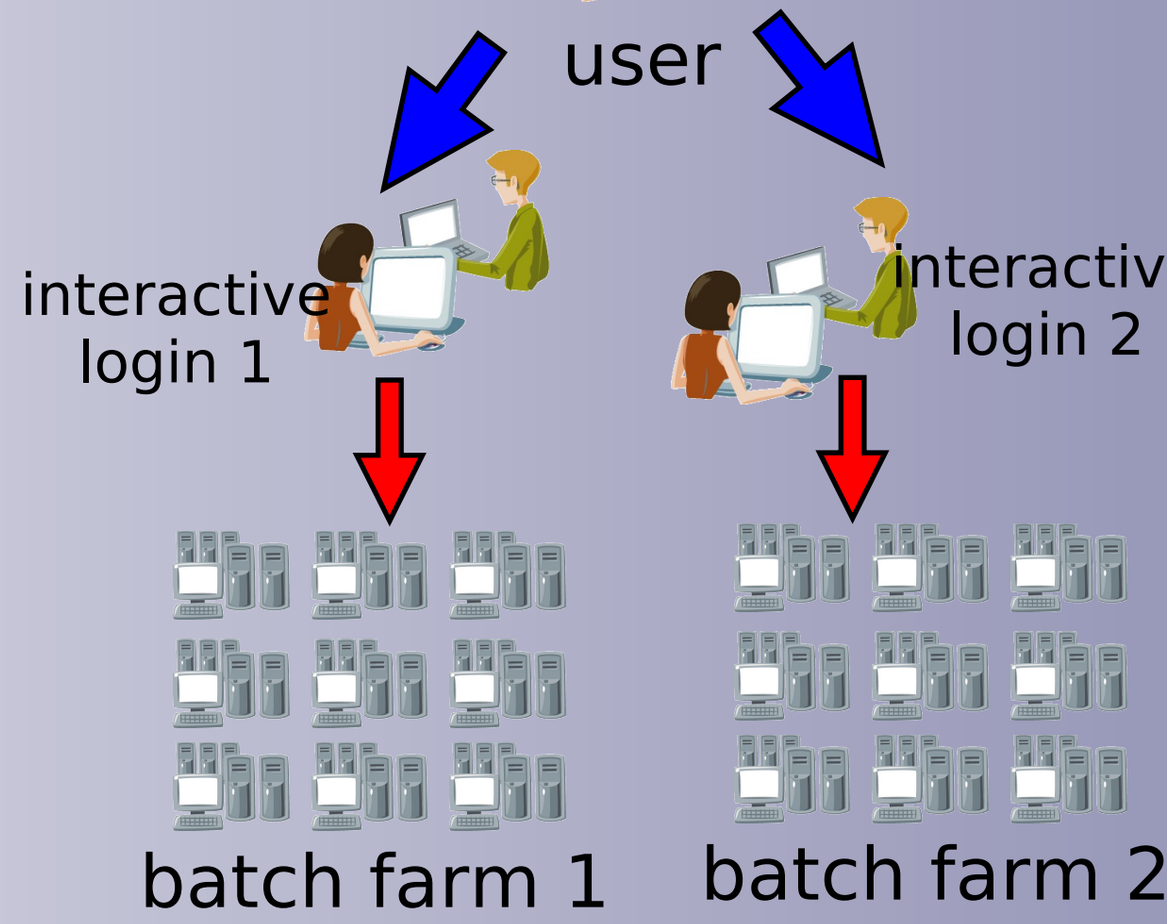
Glossary

UI	User Interface: GRID tools on your local PC
RB	Resource Broker: central job submission point
CE	Compute Element: GRID door to compute farm
SE	Storage Element: GRID door to mass storage
WN	Worker Node: Batch slot in a farm reachable through CE
EDG	European Data Grid
EGEE	European Grid to Enable E-Science
OSG	Open Science Grid
SAM	Site Availability Monitoring: continuous tests of GRID infrastructure status

Distributed computing

- Interactive analysis
  - Local user account is used to submit batch jobs to local farm
  - Administrative overhead to replicate accounts
    - User has to login to multiple machines to submit jobs
    - **Does not scale**
- Grid certificates
  - Map groups to local accounts
  - Only authenticate once for all resources





GRID certificates

- CMS uses GRID certificates and a dedicated Virtual Organization (VO) management to have better access control for specific tasks/groups
- Your GRID certificate is important, follow all rules, don’t let it expire!
- Can be a pain the first time, but then it's over

UI

- The User Interface is the GRID specific software for authentication, job submission and all other GRID interactions
- There are UI’s from EGEE and from OSG
- FNAL: pre-initialized on cmslpc.fnal.gov
- CERN initialization: ***source /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.(c)sh***
- You won't have to use it directly, CRAB uses it for you

CRAB

- CRAB enables the user to submit CMSSW jobs to all CMS datasets within the data-location driven CMS computing structure
- The aim of CRAB is to hide as much of the complexity of the GRID as possible from the end user
- CRAB provides a user front-end to
  - Split user jobs into manageable pieces
  - Transport user analysis code to the data location for execution (compiled on submitting node)
  - Execute user jobs, check status and retrieve output

Underlying analysis model

User runs interactively on small samples in the local environment to develop the analysis code and test it

Once ready the user selects a large (whole) sample to submit the very same code to analyze many more events

The results are made available to the user to be analyzed interactively to produce final plots

How to configure CRAB

CRAB is controlled by a configuration file: **crab.cfg**. This file must be in the same directory as the CMSSW .cfg file related to the analysis code.

- crab.cfg is structured into **sections,** each with related settings.

The sections are:

[CRAB]
[CMSSW]
[USER]
[EDG]

- A template crab.cfg, with comments on all the configuration parameters, is distributed with CRAB. ($CRABDIR/python/crab.cfg)

- Only some parameters are mandatory, others depend on the user's specific needs.

Basic configuration options

[CRAB]

jobtype: defines the kind of job CRAB should run (cmssw)

scheduler: defines which flavor of GRID middleware will be used by CRAB (edg,glite,glitecoll,condor_g)

server_mode & server_name: start CRAB in server mode (for larger # of jobs)

[CMSSW]

datasetpath: identifies the dataset you want to access. Enter just the datasetpath (string format /primary/process/tier) obtained from the DBS discovery page

pset : The name of the CMSSW .cfg file of your CMSSW job. This file must be in the same directory as the CRAB configuration file (crab.cfg).

total_number_of_events : Total number of events to be processed by CRAB. If set to -1, all events of the selected dataset are processed

events_per_job : Number of events per job. CRAB will create as many jobs as needed to process the total_number_of_events

outputfile : Comma-separated list of output filenames. Specify both the filename selected in the **PoolOutputSource** of the CMSSW parameter-set and/or user-specific output filenames (i.e. histogram files, txt debug files..)

datasetpath

- Check that your selected dataset exists at least at one site using the DBS discovery page
- During creation, CRAB contacts DBS to collect all information needed for job splitting and job preparation. (Number and location of files, blocks, events.)
- If needed, sites can be excluded or selected for submission (parameters in [EDG] section of crab.cfg):
  - Primary selection depends on SE name given on DBS discovery page:
    - se_black_list: exclude sites
    - se_white_list: selects sites

```
[EDG]

se_black_list = cern.ch
se_white_list = fnal.gov
```

  - Secondary selection depends on CE name given during CRAB data discovery (see submission):
    - ce_black_list: excludes CEs of sites
    - ce_white_list: selects CEs of sites

```
[EDG]

ce_black_list = cern.ch
ce_white_list = cmslcgce2.fnal.gov
```

  - “se_white_list = fnal.gov” selects all SEs which contain *fnal.gov*

Output handling

- CRAB has two ways of handling output:
  - The output sandbox
  - Copy files to a dedicated storage element (SE)
- Like the input sandbox, the output sandbox is limited in size:
  - Input Sandbox: 10 MB
  - Output Sandbox: 50 MB
    - TRUNCATED if it exceeds 50 MB ➞ corrupt files
- Rule of thumb:
  - If you would like to get CMSSW ROOT files back, please use a storage element

Basic configuration (2)

[USER]

return_data : Defines the way CRAB handles user output. Default of 1 for uses the GRID middleware sandbox.

copy_data : Output is staged out to defined SE, default is 0 to not stage out

storage_element : Defines the SE url for stage-out.

storage_path : Path on the selected SE for stage-out.

use_central_bossDB : Boss specific parameter.

use_boss_rt : Boss specific parameter.

[EDG]

rb: Defines which resource broker configuration should be used CERN or CNAF. The official configuration file is downloaded from cmsdoc.cern.ch, according to the set value (CERN/CNAF). If this parameter is commented out, the default of the used user interface is used.

lcg_version
proxy_server
virtual_organization
retry_coun
lcg_catalog_type
lfc_host
lfc_home

EGEE and catalog specific parameters

just for advanced users

Optional parameters

- ✓ copy_data
  The user can copy the output to a Storage Element of their choice.

You must specify storage element name and storage element path. Recommended for large outputs. The standard output and error from CRAB still come through your sandbox. **(Advised for CMSSW root files)**

- ✓ ui_working_dir
  The user can specify a working directory name different from the default convention.

- ✓ se_white_list/se_back_list
  Control at which GRID site the jobs **will/will not run**, using the list of storage elements (from the discovery service). You can define sites which the job should/should not use to run your jobs. May also do site selection by using additional compute element criteria

See the CRAB documentation for the complete parameter list and descriptions

How to run CRAB

The basic CRAB workflow is organized in 4 steps:

- Job Creation
- Job Submission
- Job Check status
- Output retrieval

Job Creation: **crab –create**

At this level CRAB interacts with the DBS system, organizes the jobs of the task according to the user's job splitting parameters, packs the users specific code(/lib /module /data), prepares the script to configure the remote environment, and (using BOSS) prepares the jdl file to communicate with the RB

It also creates the working directory which is organized in 4 subdirectories named:

/job : CRAB specific stuff
/log : CRAB log file location
/res : default results destination
/share : CRAB and BOSS specific stuff

How to run CRAB (2)

Job submission: **crab –submit**

The submission uses the previously created CRAB project to submit the jobs.
Before the real submission, CRAB always checks for available resources preventing the submission of unmatched jobs. By default all created jobs are submitted.

Job status: **crab –status**

This command checks the status of all jobs in the CRAB project.
For each job CRAB prints on the screen the job id, scheduler status, site hosting the jobs, cmssw exit code, & job exit code. The output gives also a summary with a list of job IDs sorted by status categories.By default the status of all jobs is checked.

Job output : **crab –getoutput**

This command retrieves the output of all jobs of a CRAB project which have status “Done”. By default the retrieved output files are copied in the “res” sub-dir of the CRAB workingdir. Included are the standard output and error of the jobs (CMSSW stdout and stderr) and the output files specified in crab.cfg.

Even if your job fails, run **crab -getoutput**. Otherwise your output clogs up a server.

How to run CRAB (3)

For submit, status and getoutput the user can override the default behavior selecting individual jobs by:

- All : default behavior
- 1,2,3 : individual jobs
- 1-3 : jobs range

The latter two can be combined using commas (e.g. 1,2-4)

Using a specific directory for a CRAB project each command must be executed as:

crab –<command> -c <directory>

Tutorial

- Time to put this into action

https://twiki.cern.ch/twiki/bin/view/Main/EricVaanderingCRABTutorialJun2008

How to get CRAB support

Best source for user support is the CRAB feedback hypernews:
https://hypernews.cern.ch/HyperNews/CMS/get/crabFeedback

All CRAB questions and suggestions can be posted to this forum, CRAB developers try to solve the problems and give solutions. User suggestions can help improve the tool.

A troubleshooting guide is at
https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookGridJobDiagnosisTemplate
(still under construction)

Questions not directly related to CRAB (GRID related problems, CMSSW specific problems, etc...) should be referred to other hypernews forums

Additional Documentation:

Web Page: http://cmsdoc.cern.ch/cms/ccs/wm/www/Crab/
Wiki: https://twiki.cern.ch/twiki/bin/view/CMS/CRAB

Backup Slides

Additional CRAB features

- ✓ crab –testJdl

  Check if there are compatible resources where the job can run.

  crab query the resource broker to know if sites which have the samples are accepting jobs

- ✓ crab -kill

  Kill jobs which have been submitted (incorrectly) to the grid.

- ✓ crab -resubmit

  Resubmit exactly the same job (which is either abort or killed)

- ✓ crab –printId

  Find the GRID id of specific jobs (can be useful for debugging purposes)

- ✓ crab -postMortem

  Find more info about aborted jobs which may help in debugging problems.

DBS

- Data Bookkeeping Service
- What datasets exist where. What files they contain and mapping to runs, lumi blocks
- Production creates datasets and registers them with DBS

- Dataset is PrimaryDataset/ProcessedDataset/Tier
  - Primary: Describes the physics channel
  - Processed: Which software was used to process
  - Tier: Kind of information: RAW/SIM/DIGI/RECO/etc
- Homepage for DBS is http://cmsweb.cern.ch/dbs_discovery/

More on DBS

- DBS has a number of ways to find data, will only cover the common ones
- Navigator (default)
  - Series of pulldown menus (CMSSW version, event content, physics process, and more)
  - Choices further restricted as choices are made
- Site search: Find out what datasets are available at your local Tier2
- RSS feeds: Like a podcast for CMS data
- Much more info and a tour:
  https://twiki.cern.ch/twiki/pub/Main/CRABatFNALTutorialJune2007/DBS2_discovery_FNAL.pdf